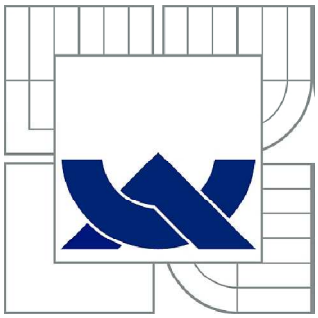
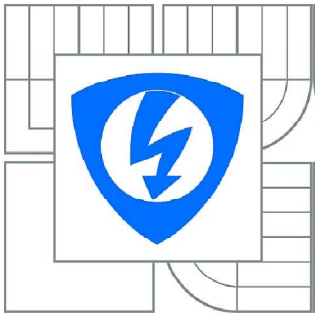


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

VYUŽITÍ ETHERNET POWERLINKU PRO SPRÁVU PERIFERIÍ V REÁLNÉM ČASE

REAL TIME ADMINISTRATION OF PERIPHERALS USING ETHERNET POWERLINK

BAKALÁRSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

IKER JUANENA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAKUB DOKOUPIL

BRNO 2011

Bachelor's thesis

bachelor's study field

Student: Iker Juanena Sánchez **ID:** 137123
Year of study: 1 **Academic year:** 2010/2011

TITLE OF THESIS:

Real-time administration of peripherals using Ethernet Powerlink.

INSTRUCTION:
Implement communication based on Ethernet Powerlink into PC workstation. Create real-time communication between PC workstation and distributed peripherals of B&R using X20 Bus Controller. Verify real-time properties of Ethernet Powerlink for the transfer of information between individual peripherals for various levels of sampling periods. Create development environment allowing an implementation of control algorithms for the specific hardware configuration with X20 Bus Controller. Verify theirs functionality on a real process.

REFERENCE:
http://www.kalycito.com
www.ethernet-powerlink.org
Zurawski. R: The industrial communication technology handbook, Florida: CRC Press, 2005. ISBN
978-0849330773
Ogata. K: Modern control engineering, fourth edition. Upper Saddle River, New Jersey: Prentice Hall,
1997. ISBN 0-13-314899-8

Termín zadání: 7.2.2011 **Termín odevzdání:** 30.5.2011

Head of thesis: Ing. Jakub Dokoupil
Předseda oborové rady

Abstract

Nowadays the global production of anything in the industry is manufactured by automatic processes. It is essential to keep under control any automated process in the industry to ensure the proper development of the factories. The most important thing for this aim is to have everything properly connected and to have a reliable communication system. Over of the course of the last two decades it has been a demand that has become more pressing for a reliable communication system to fit out a high quality communication in real time. Today there are a lot of reliable and high quality communication protocols even the ones who ensures a real time communication but the most interesting protocols are the ones that less cost entail in our factory. In this way they achieve an adequate efficiency. This project gives an overview of how to build a reliable, cheap and real time communication system.

Bibliographic citation of my Thesis

JUANENA SÁNCHEZ, I. *Využití Ethernet Powerlinku pro správu periferií v reálném čase.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. XY s. Vedoucí bakalářské práce Ing. Jakub Dokoupil.

Declaration

I hereby confirm that I worked out this project myself with the aid of my thesis leader, documentation and other sources of information that are mentioned at the end of the thesis.

As the author of the thesis, I declared that the information in this document has been carefully checked and is believed to be entirely reliable. However, the author of this thesis assumes no responsibility for any inaccuracies. This thesis author neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product.

In this manual are descriptions for copyrighted products which are not explicitly indicated as such. The absence of the trademark (©) symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

In Brno the: **30th of May of 2011**

.....
Author sign

.

Acknowledgements

This project would not have been possible without the help and support of the following people.

Firts off all, I would like to thank and express my appreciation to my project supervisor in Czech Republic Ing. Jakub Dokoupil and thank you to all the department as well. He has patiently explained me all that I needed to develop this project. I really appreciate your help. Secondly I would like to thank the people who works for the sourceforce web site because they have a brilliant help forum which works fantastically. They have patiently explained me all that I needed to develop this project.

Finally, I want to thank my parents as well, because I wouln't be here developing this project in the Czech Republic if they had not been so helpfull and encouraging as they are.

Contents

List of Figures

List of Tables

1. Introduction

The last course 2009/2010 I was suggested a project to complete my Bachelor degree and moreover I was suggested to make it abroad, specifically in Brno, Czech Republic. My bachelor thesis is basically the implementation of a real-time communication between a PC workstation and a B&R manufactured Bus Controller, based on a predefined communication protocol, which is Ethernet POWERLINK. Actually a communication protocol is a sequence of rules that the devices have, to understand each other, to communicate over a network. There are currently many different protocols, but the concrete application of the factory will limit the users to choose the proper one. Some are free of charge and there are also private protocols, so here it is another important point. This project is developed with Ethernet POWERLINK, a protocol completely free and available to everyone on the internet.

In despite of being a project with academic objectives, the usefulness of all the knowledge learned and the system itself is clearly applicable to industrial and engineering environment around us. Thus, this report aims to collect such knowledge during these months. This report also contains a CD where the user can find all the necessary projects to set up the real time communication and also contains some audiovisual data such as pictures and one video making a demonstration. The programs they also stated that all it is said in this document has been done and even more, that it has been proved and it worked.

1.1 Precedents and Problems

Communication protocols exist since they began to connect computers to each other creating a Local Area Network (LAN). It was completely necessary to have a protocol to be able to set a communication between them. The original manufacturers of protocols they made their own designs that were subsequently either standardized or disappeared, depending on the acceptance they had in the market. Basically the different network standards differ from their topology or interconnection of the nodes and the way of the media access. The mean which the network is connected also is very important to determine the features offered by the protocol.

One of the main problems was to ensure a suitable response time to give the choice to all nodes to use the media when they really needed, before a reasonable time has elapsed. This feature is knows as determinism. In a system that requires real-time communication is essential to have a deterministic protocol or at least a part of the protocol has to be deterministic to ensure the proper updating of all the variables before a reasonable time has elapsed. This communication might be complicated due to the collisions in the communications, for instance, when two or more devices want to transfer data to the mean at the same time and also we want to keep everything update in the same moment. In this case and in some protocols there might be collisions so the communication would be impossible.

1.2 Aims of the project

In order to solution the problems we have set out in the last section and in order to create a real-time communication between a PC workstation and another device, a Bus Controller developed by the B&R company in our case, Ethernet POWERLINK protocol is going to be used. Hence, the main objective of the project has been the adaptation and configuration of the software and the hardware to create a real-time communication in our system.

1.3 Phases of the project

All the phases of this project they appear in a Gant Diagram attached in the appendix of the document (Figure 34) and (Figure 35).

2. Development of the project

The first stage of this project has been to establish a sequence of steps to finish the project, where the first one has been the one call documentation phase. In this first step the main aim has been to get information about the protocol used in this project. This protocol is the one said before, which is Ethernet POWERLINK and this will be in charge of computer and Bus Controller communication.

This protocol comes up because there was a great demand of more reliable communications system that would offer high flexibility across the board compatibility. Ethernet POWERLINK predecessors had not achieved but in the end a combination between some protocols and a big effort from the manufactures resulted in something called Standard Ethernet. In order to develop Ethernet-based, but real-time capable field buses, manufactures have pursued various approaches in their efforts to eliminate such delays. These solutions are commonly referred to as like Industrial Ethernet and among them the one which more reputation had been achieved has been Ethernet POWERLINK.

This paper will explore the basics of this protocol as well as its main features.

2.1 Introduction to Powerlink

The main feature of this is that everything is connected through the same mean, thus avoiding gateways, routers and etcetera. This means that we create a single network for all systems (Figure 1). This feature provides user a single, consistent and effective way to enable all type of communications in modern automation. A POWERLINK network integrates all kind of modern automation devices, including the one used in this project. Specifically the device we use in this project is an input/output module, developed by B&R. It is not exactly a PLC but it enables us to create a real-time communication with the computer.

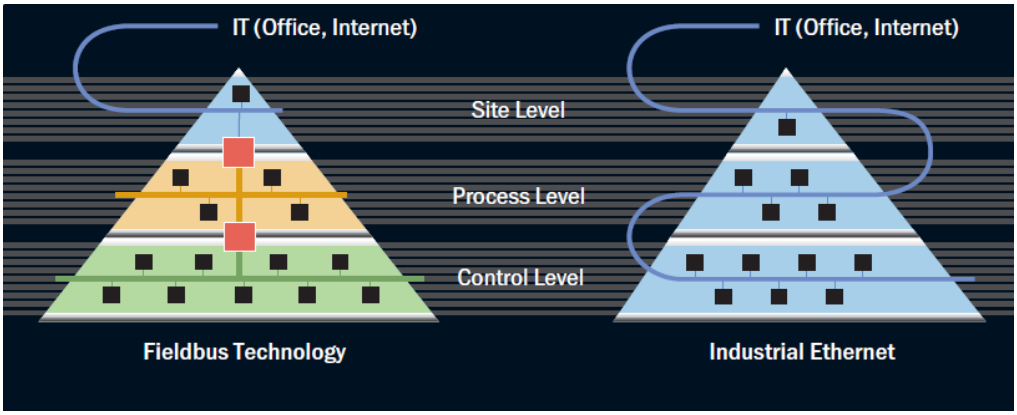


Figure 1: Fieldbus Network - Ethernet POWERLINK network [1]

Due to its features POWERLINK is an Industrial Ethernet solution devised to give users a single, consistent and integrated means for handling all communication tasks in modern automation. Moreover allows the cross data traffic and also gives the user a completely open choice of the topology of the network. In one hand the permission of the cross traffic means that all devices can deliver data transfers directly to the network while the devices that should read can read it whenever. In the other hand the open choice of the topology will allow the user to modify or expand the network without any problem.

Unlike other Industrial Ethernet systems, POWERLINK is a purely software based solution. Hence POWERLINK is one of the expanded protocols of the world. Nowadays is used in thousands of sectors in the Industry like in machine and plant engineering, in process automation and measuring technology.

2.1.1 The Mechanism

As mentioned previously, Ethernet POWERLINK is a solution 100% based on the software that is in line with the IEEE 802.3 Ethernet Standard [1]. As well as this, it is a protocol that allows us to communicate in real time. Ethernet POWERLINK protocol communication proceeds as it does in an organized roundtable conversation, where a moderator prompts participants to make their statements. In this scenario the moderator sees to it that everyone gets a turn to get a world in inviting the participants

one after the other to speak at a specific time. In contrast to standard Ethernet, this procedure will ensure that different nodes do not talk at the same time.

A system based on a POWERLINK network would have the following structure of communication: On one hand one node arbitrary designated, according to one criteria is going to have the function of the conversation moderator and this node is going to call the Managing Node (MN). In the other hand all other devices operate as Controlled Nodes (CN). The MN sets the clock pulse to synchronize all other units and also manages the data communication cycle (Figure 2).

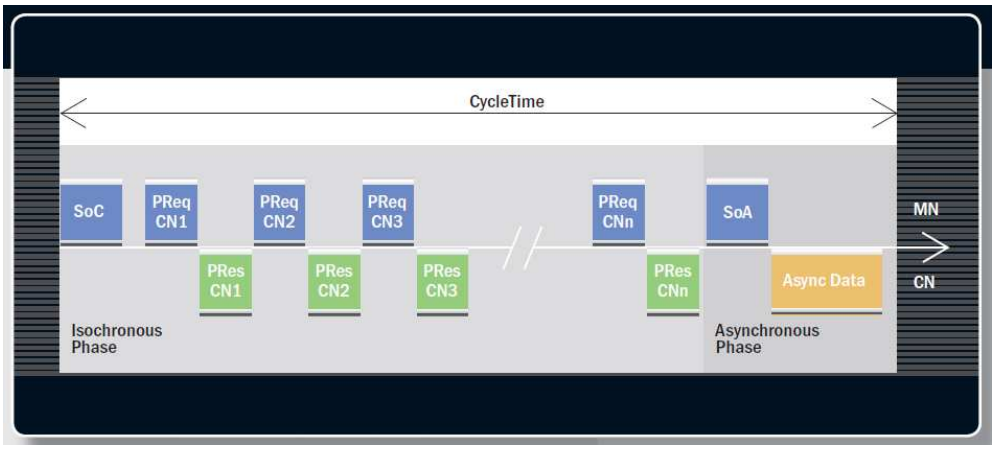


Figure 2: : Data communication cycle diagram[1]

In the course of one clock cycle the MN sends its so-called Poll Requests to all CNs one after the other. They reply immediately to these prompts with poll responses. POWERLINK cycle consists of three periods. During the start period the MN sends frames called “Start of Cycle Frame” (SoC) to all CNs to synchronize the devices. The exchange of the information comes on the second phase, isochronous phase. In this phase, the protocol behaves as a deterministic protocol, which means, that is the time for transferring the information that must be controlled within a reasonable time. And finally the third period of a cycle marks the beginning of the asynchronous phase, which allows the transfer of the not time critical data packets.

The mechanism of the communication cycle explained before (Figure 2) is exactly one single cycle. This cycle time in Ethernet POWERLINK is 100μs and the system is able to connect up to 240 different nodes due to the bandwidth 100Mbit/s. The bandwidth is closely related with the mean the network is connected. In this project will be a common Ethernet wire with two RJ45 connectors.

2.1.2 Main characteristic

2.1.2.1 Topology

One of POWERLINK most important characteristic is that the user is 100% free to choose any type of network topology. Is understood as network topology the way we connect the nodes into the network. For instance we can connect the nodes one after the other creating a daisy chain, or we can connect the nodes creating a star topology. This feature is very useful if we want to modify the distribution in our company or we want to expand it with more nodes, because these changes can be made at any time and in any way without compromising the application.

2.1.2.2 Pluggins

This characteristic means that the user can plug the nodes in or disconnect from the network at any time, without compromising the functionality of the system. There is no need to restart the system for become operational again. Thanks to this function the user can save a lot of time.

2.1.2.3 Direct cross traffic

Ethernet POWERLINK protocol works according to the broadcast scheme. This means that on one hand all nodes are free to broadcast the information in the network and moreover all nodes can receive this broadcast. This is possible because the nodes can tell by the target addresses on data packets whether any such data is intended for them.

2.1.2.4 Multiplexing

Multiplexing is essential to optimize the bandwidth used in the isochronous phase. There are situations where not all the nodes have to be polled in every cycle. These gaps let the protocol to multiplex other nodes.

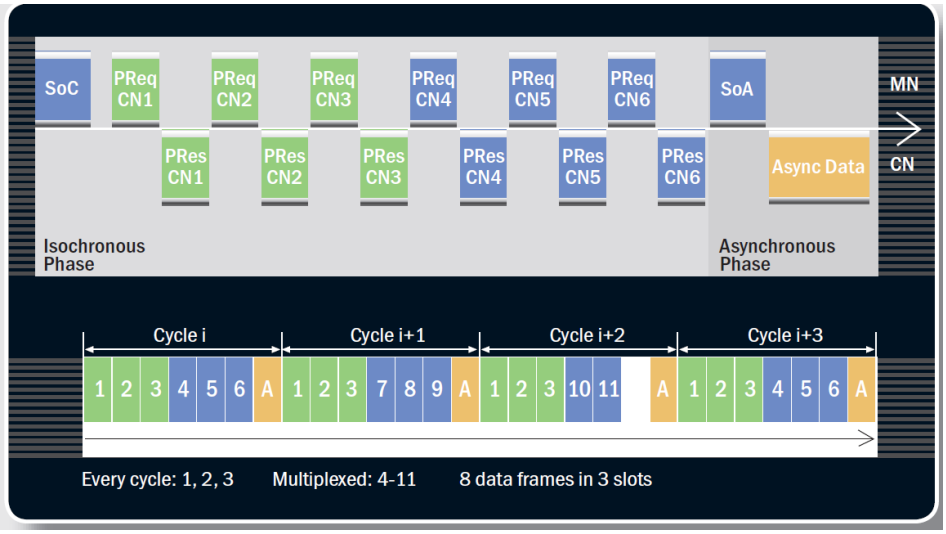


Figure 3: Multiplexed communication cycle diagram[1]

We can see (Figure 3) that in these 4 different cycles there are nodes polled every cycle and there are other ones that they are polled every three cycles. For instance the nodes 4, 5 and 6 are polled every three cycles.

2.1.2.5 Asynchronous data

The asynchronous phase is the last phase of the cycle. It is reserved for non time critical communications. In this phase, data packets can be transmitted in standard Ethernet frames. If a device has more such data to send than a single cycles periods has capacity for, the transfer is spread out over the asynchronous phases of several cycles. Data of any kind can be transmitted in this phase. Just say that Service Data Object (SDO) for device configuration and diagnostic, for example are transmitted in this way.

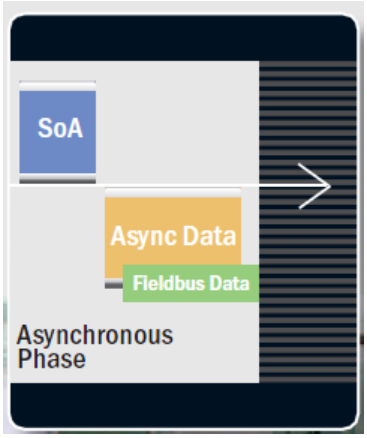


Figure 4: Asynchronous data[1]

2.1.2.6 Redundancy

Redundancy is a security method which is device to act as soon as any failure is detected in the system. POWERLINK enables the use of two types of redundancy: Ring redundancy and Master redundancy.

The Ring redundancy is a very cheap and simple option to ensure an immediate response if one of the wires is broken. Applications are linked in the form of a ring; both outer ends of the data line running through the network are connected to the controller (Figure 5). This type of redundancy is typically used for solutions exposed to considerable mechanical stress.

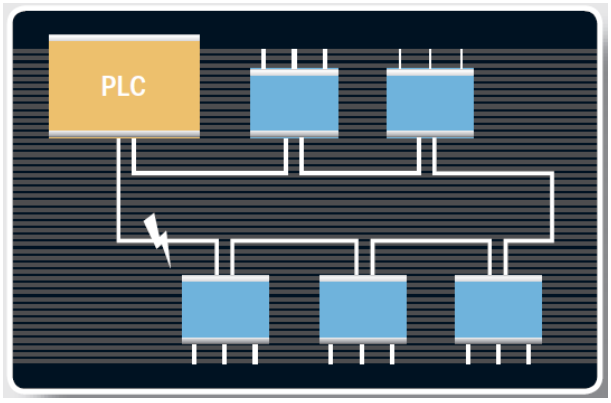


Figure 5: Ring redundancy distribution[1]

The Master redundancy is used in energy systems within the process industry. Mater redundancy is based on two or more redundant Managing Nodes at the top of the network. Only one of them serves like Managing Node while the other ones are acting like Controlled Nodes despite that they are ready to start working like a Managing Node if the main node fails. This redundancy model allows for a wide range of topologies.

2.1.2.7 Powerlink safety

Reliably recognizing all data transmission errors, the open protocol achieves a high degree of safety. This means that the protocol is able to recognize all the transmissions errors and not only recognize but it is able to sort out the faulty data transmissions.

2.1.2.8 CANopen

The CANopen and POWERLINK protocols are similar protocols because POWERLINK has adapted the communication profile of the CANopen. One of the mayor decisions made by the Ethernet POWERLINK Standardization Group (EPSG) was to define the protocols application layer as a carrier of all CANopen mechanisms. In this way Ethernet POWERLINK has become a CANopen communication profile predecessor. CANopen is still, one of the most widely used application protocols today. POWERLINK also uses the same device description files as CANopen and the same Object Dictionaries. The main advantage of this is that devices from different manufactures that share the same description files can easily replace each other with no need for any reconfiguration.

Nowadays the application complexity is generally increasing, which leads to a greater number of nodes, rising data load, and higher performance requirements. A simple migration to POWERLINK is an ideal alternative for users who wish to continue enjoying CANopens benefits.

This document will go through the CANopen application layer briefly in the next paragraphs, in this way, the user will have an idea about what is the application layer.

CANopen Application Layer, which has imported the Ethernet POWERLINK protocol, is composed by the next four services [2].

1) CMS (CAN base Message Specification) which offers variable type objects, event domain to design and specify how to access the functionality of a device via CAN interface.

2) NMT (Network Management): provides services for network management. It performs the tasks to initialize, start, stop or detect node failures. It is based on the concept of master-slave, having only one NMT master in the network.

3)DBT (Distributor): Is in charge of assigning all the dynamic CAN identifiers (11bits) or in another way called the COB-ID (Communication Objetc Identifier). It is also based on the concept of master and slave having only one DBT master.

4)LMT (Layer Management): Allows to change certain parameters on the layers such a the node identifier (Node-ID) or the speed of the Bus.

2.2 Controlled node device description

The second step of this project has been to analyze the specific device we are using in this project as a Controlled Node. This step is essential for setting up the network because we have to know all the small features of the Bus Controller and the Input / Output modules before starting programming.

In our case these are the main characteristics of the controlled node used in this project. We are going to use an Input / Output device composed by the next modules: First of all, the main device is the X20 family Bus Controller which makes possible to connect Input / Output nodes to POWERLINK network. This device is exactly, X20 BC0083 (Figure 6).

This is going to be in charge of the communication because it makes possible the data exchange between the Input / Output modules and the XP computer, through the POWERLNK network. The device has in his front some switches to set the Node ID. This step is going to be really important to establish a connection with the computer because later in this document this Node ID parameter is used. It has two switches because is it possible to set until 255 different controlled nodes in the Powerlink network. In this project the Node ID is set to 01. It works with hexadecimals numbers and be careful because some addresses are occupied for other aims (Table 1).

Figure 6: X20 BC0083[3]

Switch Position	Description
$00	Reserved, switch position is not permitted.
$01-$EF	Station number for POWERLINK station. Operation as a controlled node.
$F0-$FF	Reserved, switch position is not permitted.

Table 1: BC0083 Station numbers [3]

Attached to the Bus Controller, as every device needs power supply, it also needs, so the next card of the Controlled Node is exactly, X20 PS9400 which works together with the X20 bus controller. It is equipped with a feed for the bus controller, the X2X Link, and the internal I/O supply. The X2X link is the link the controlled node has inside to communicate with the Input / Output modules. Then we have all the Input and the Output cards consecutively (Figure 7). They are exactly X20 DO9321, X20 DI2377, X20 AI2622 and X20 AO4632. The first hard, the DO9372 module is equipped with twelve outputs for one wire connections. After that the second card, which is exactly the DI2377 module, it is equipped with two inputs for three wire connections. Both inputs can be configured as event counters. Consequently the third card, AI2622 is equipped with two analog inputs with 12 bit digital converter resolution. Finally the last card in the application is the AO4632 which is equipped with four outputs with 16 bit digital converter resolution. In this project two different process are controlled and each card is use in the proper application. For instance to control an analog system it is compulsory to use the analog I/O cards. However you can use it as well in order to control other systems but it is no so common. On the other hand the digital I/O cards they are used to control a digital actuator like sensors or relays. Usually the relays they are used to control a high power systems.

The next three pictures describe as good as possible how it looks the device after been assembled (Figure 7).

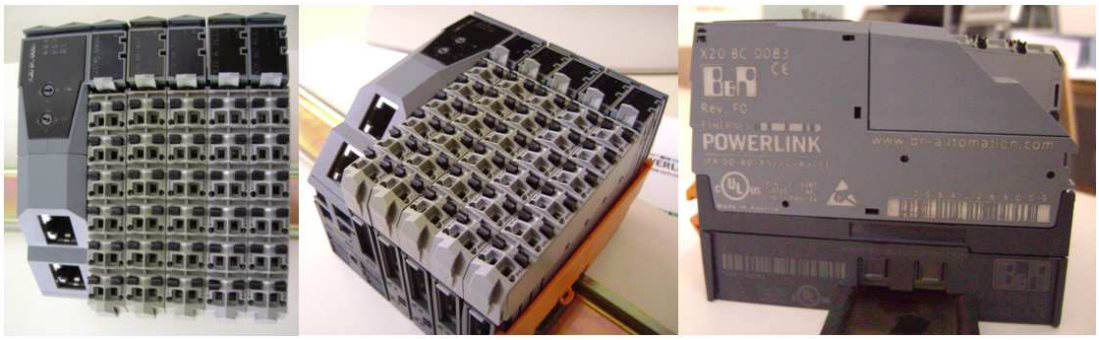


Figure 7: Controlled Node with all the I/O modules

2.2.1 Electric connection

Related with the electric connections the user will find some information in this report below but for further help is essential to check the datasheet of each device. The datasheets of each device have something similar to this pin assignment (Figure 8). This one is exactly from the X20 AI2622 but all of them look really similar. It is really important to make the connections properly to avoid any problem with the hardware.

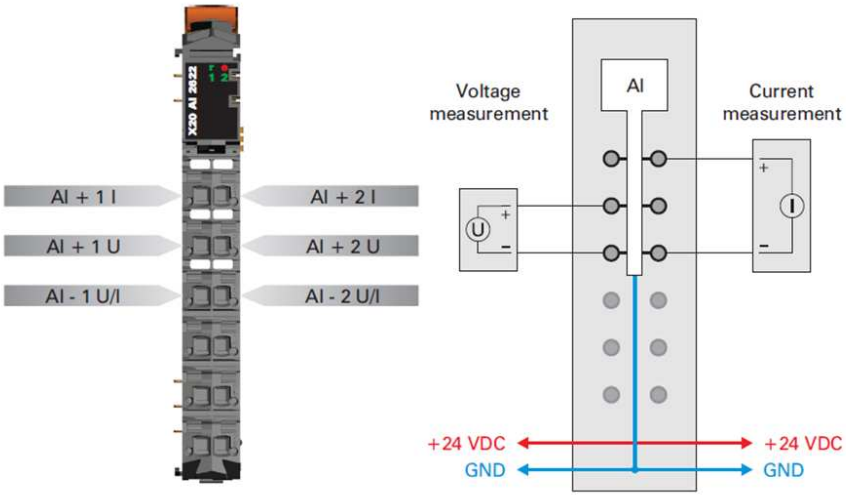


Figure 8: Pin Assignment X20 AI2622[4]

2.3 Get and Install all the necessary software

Once we know, which are the main features of the protocol we are using and which is our controlled node, the third step is to get and install all the necessary software. As is said before POWERLINK is 100% open source technology so we can find all the necessary documents and programs on the internet. OpenPOWERLINK is in this case the name of the industrial Ethernet solution, programmed in ANSI-C [5] which enables us to set up a system and putting it into operation.

IMPORTANT: OpenPOWERLINK is an open source technology but remember that, if we are building the project with Windows XP we will need the license and the same happens with Microsoft Visual Studio, the programming environment. The Automation Studio FieldbusDESIGNER is also private software which needs a license, but in this case we can use a trial version that expires in 30 days. There is also the possibility to build the network in Linux which means that, it wouldn't have so many expenses. This document will show you how to build the network using Windows XP.

We will have three main programs and then some other plugings and drivers to have everything already. This report will explain all of them:

2.3.1 B&R Automation Studio FieldbusDESIGNER V3.0.80

B&R FieldbusDESIGNER allows convenient and professional configuration of B&R's Input / Output modules and fieldbus components. The program will create the xdc file of our hardware, which is the description file of it. B&R FieldbusDESIGNER provides access to all available parameters, both on the bus controller as well as on the I/O modules. The configuration files generated will be used in other solutions, such as the POWERLINK openCONFIGURATOR.

We can download from: [6] and then follow the instructions of the webside to install it properly.

2.3.2 OpenCONFIGURATOR V1.2.0

OpenCONFIGURATOR is an open-source configuration tool for easy setup, configuration and maintenance of any POWERLINK network. OpenCONFIGURATOR ideally complements openPOWERLINK because thanks to this tool we can create the necessary files for running a POWERLINK network. The tool will need the xdc file created with B&R FieldbusDESIGNER to be executed properly and then it will create other two files to continue with the setting up of the network.

We can download this software clicking on the address [7]. Remember that we will not need any license. The installation of the tool is really easy, there are also help documets on the webside also there is the possibility to open the manual directly from the tool, help. To run the program is necessary to install a driver called ActiveTCL 8.5.9.1, also available on the internet free of charge following this link [8]. It is really important to ensure which proccesor we have (32 bits X86 or 64bits X64) to install the correct driver. If we do not know which structure has our computer is it possible to check it following this instructions: Start->All programs->Accesories->System Tools->System Information.

2.3.3 Microsoft Visual Studio 2005 or newer version

Finally it is necessary to have in our computer this program because is going to be the programming environment. Thanks to this tool we will be able to change the execution code of our network. This program as is mentioned before needs license. This license can be bought on the internet or in any specific shop.
Once Microsoft Visual Studio is installed, it is also necessary to install two drivers more:

1) WinPcap driver, which consists of an application programming interface (API) for capturing network traffic. Our network communication will work in real time so that is why is essential to have this driver. We can find this driver on the internet absolutely for free following this link [9].

2) Dotnetfx35 driver, is a software framework that runs on Microsoft Windows and it allows a quicker development of the aplications. Download from [10].

2.3.4 OpenPOWERLINK

OpenPOWERLINK is an open source industrial Ethernet solution provided by SYSTEC electronic and it is basically the last piece we need to start with the setting up of the network. It contains the Ethernet POWERLINK protocol stack for the Managing Node (Master) and for the Controlled Nodes (Slaves). Here there is all the Application Programming Interface (API) which has to be familiar with us because here is going to be where we are going to set up our system.

To obtain this folder called openPOWERLINK V1.7 we have to follow this link [11]. All we have to do is to donwloaded and keep it in a safe place of the memory of our computer. Then, the document will make an explanation about what contains this folder.

2.4 Setting up of the network

If you are at this point of the document it is supposed that you have already all the neccesary programs and documents to start buiding the network. The document is going to go through the different programs explaining what the user must do.

2.4.1 FieldbusDESIGNER configuration

This section provides assistance for configuring the B&R POWERLINK bus controller. The use of this tool is very easy because the programming interface of the program is very visual. The only thing that the user has to do is to specify one by one all the cards the Control Node has and to set up some variables. The process of building a project will be divided in five steps:

2.4.1.1 Creating a project

Once the program has been installed and opened the first step is to create a project. The project can be created by **File / New Project** menu (Figure 9).

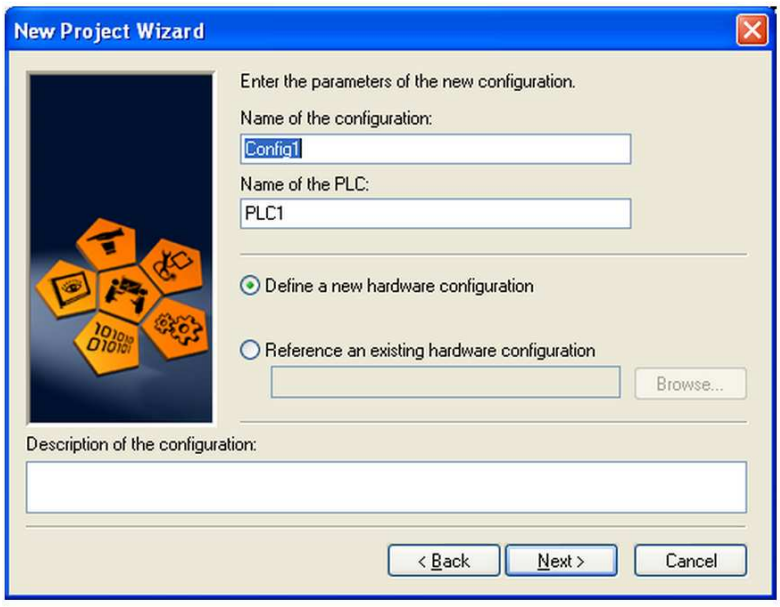


Figure 9: FieldBus DESIGNERNer Project Window

After clicking **next** a pop-up appears (Figure 10) to select the CPU. The CPU type used in this project is POWERLINK CPU.

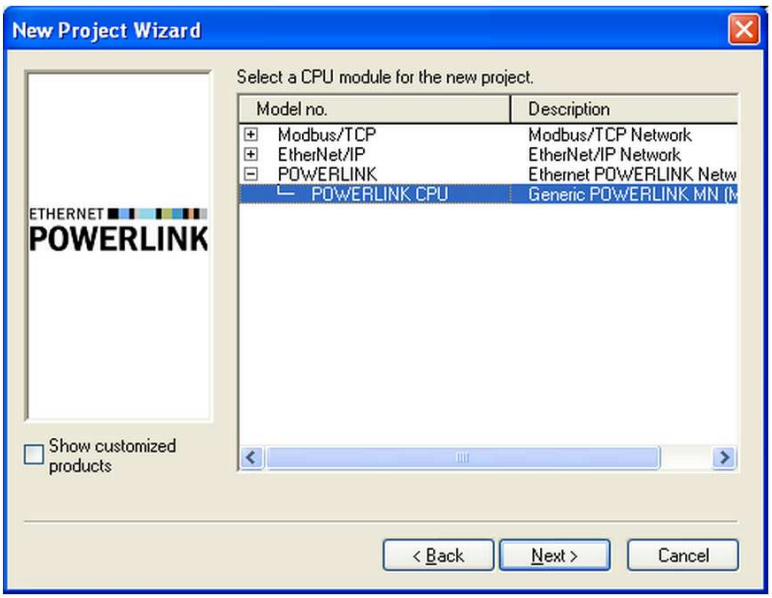


Figure 10: FieldBus DESIGNERNer CPU selecting window

And finally a small summary of the project is shown to check all the information. Then we click **Finish** to close this pop up. Finally the project is created.

2.4.1.2 Adding and configuring the Bus Controller

Once the project is created the user will see a window where the CPU we have chosen will appear. The next step is to add a Bus controller and for this aim we should open the communication interface of our POWERLINK CPU clicking with the right button as shown in the (Figure 11) and after selecting **Open POWERLINK**.

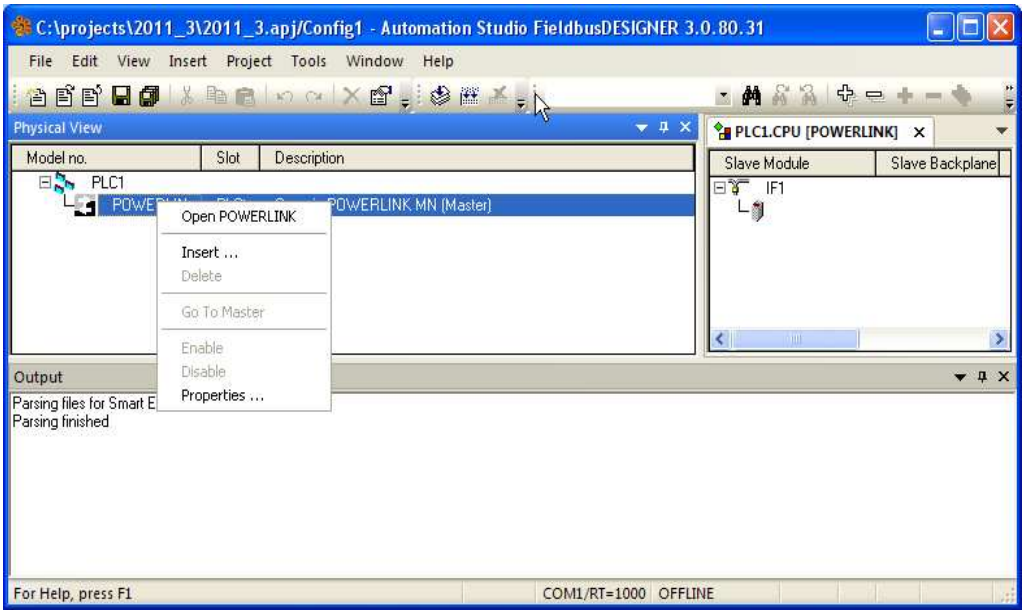


Figure 11: FieldBus DESIGNERNer Physical View

A POWERLINK bus controller can be added using the **Insert** option in the shortcut menu for interface IF1. The bus controller used in this project is the X20BC0083. Click **next** after selecting it (Figure 12).

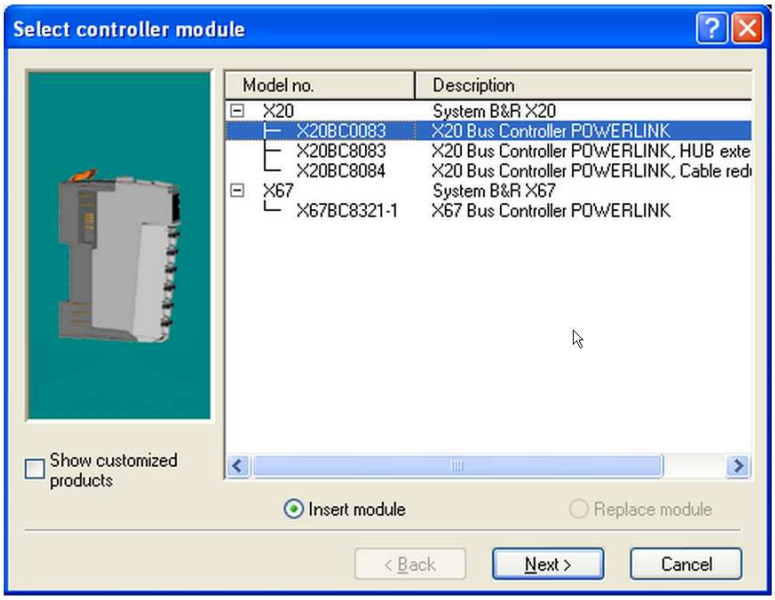


Figure 12: FieldBus DESIGNERNer Bus controller selection

The next window (Figure 13) is to set some variables: The first one is the node number, which means the number of the node in our network. It can be set by some switches in the physical hardware. The node number used in the project is set to **1**. The second one is a gap, to have an opportunity to change the name of our hardware module. The third one is to specify if our node is going to be multiplexed. In this case there is only one node so it isn't multiplexed. And the last one specifies that the X2X bus into the bus controller, is synchronized with POWERLINK cycle with the Start of Cyclic (SoC). The X2X bus is the bus that the Controlled node has inside to communicate with the rest of cards, for instance digital input and digital output cards.

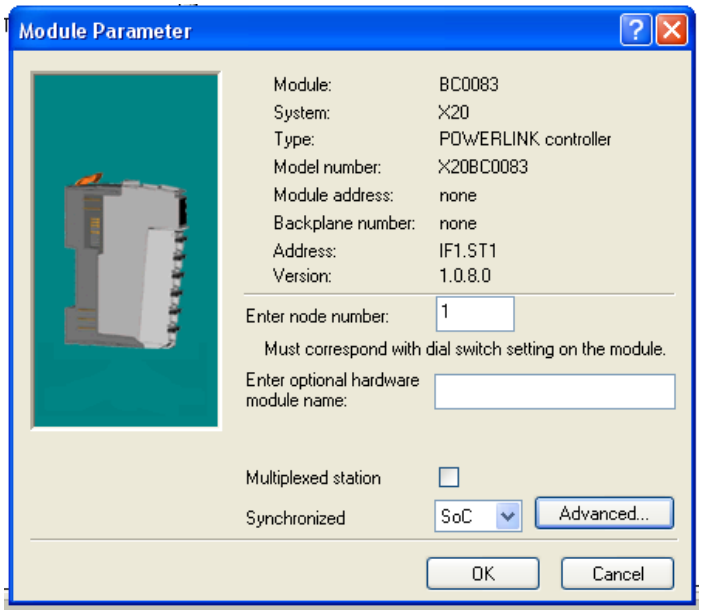


Figure 13: FieldBus DESIGNERNer Module Parameter

After clicking **OK** the Bus Controller is added to the project. By default this bus controller brings his own power supply that is exactly X20PS9400. We can check this on the physical view of the project.

2.4.1.3 Configuring Powerlink network properties

The next step is to set up the POWERLINK network properties (Figure 14). This window can be opened clicking with the right button on IF1 and after **properties**, but the one is on the <project_name>/ CPU [POWERLINK] tab, so far the only one you should have.

Figure 14: FieldBus DESIGNERNer POWERLINK Properties

The next pop-up appears. The cycle time specifies the length of a POWERLINK cycle. In this project the cycle time is 10000 µs because is a suitable cycle time for is aim. The Prescaler gap is to set the Number of POWERLINK cycles per multiplexed cycle. This value is set by default but as the user didn't select the multiplexed station (Figure 13) nothing happens.

IMPORTANT: In this POWERLINK cycle time, doesn't matter which value the user has putted because after the openCONFIGURATOR will overwrite the POWERLINK cycle time value.

2.4.1.4 Adding I/O modules and configuring the X2X properties

Once the bus controller is added to the network, the rest of the cards should be added to the network as well. To insert I/O modules, the bus controller's X2X interface must be opened by selecting Open X2X Link from the physical view. To do this, right click on the bus controller and then click on **Open X2X Link**.

A new tab, the second one, is opened in the right section of the Fieldbus DESIGNER. After that, a new I/O module can be inserted in an empty slot by right clicking and then selecting **Insert** in the shortcut menu as shown in the (Figure 15).

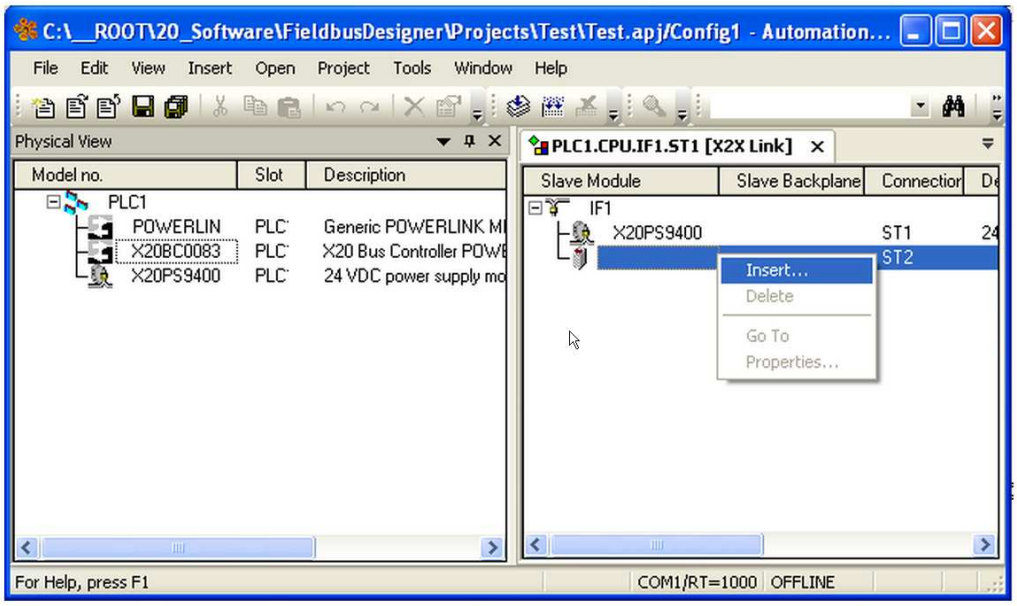


Figure 15: FieldBus DESIGNERNer Inserting new modules

After that a new window opens (Figure 16) where all the available modules appear. All this modules can be added to our project. This document will show you how to add one

module. For a complete configuration it is necessary to repeat the same process to add all the modules.

IMPORTANT: It is essential to add the modules exactly in the same way that there are on the real hardware.

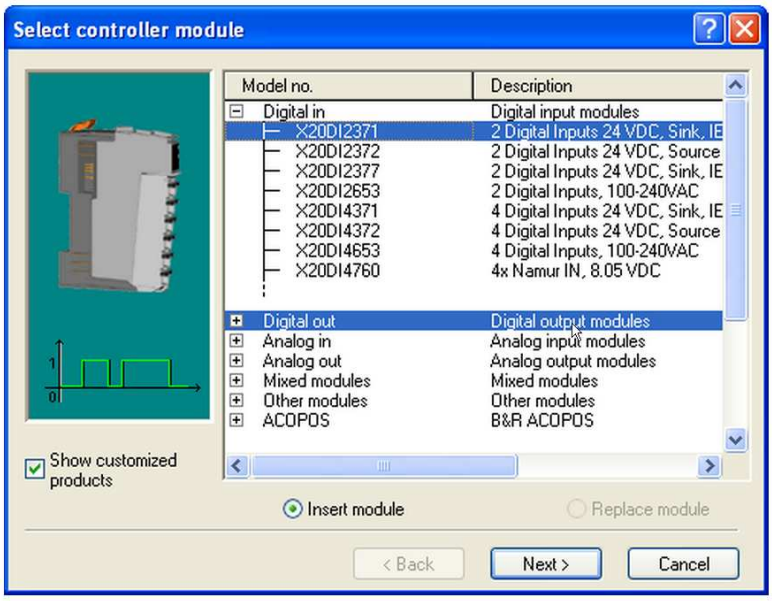


Figure 16: FieldBus DESIGNERNer: Available modules

All it is necessary to do is to select the desired module and click next. Each module has its own configuration window (Figure 17) that will appear after executing the previous step.

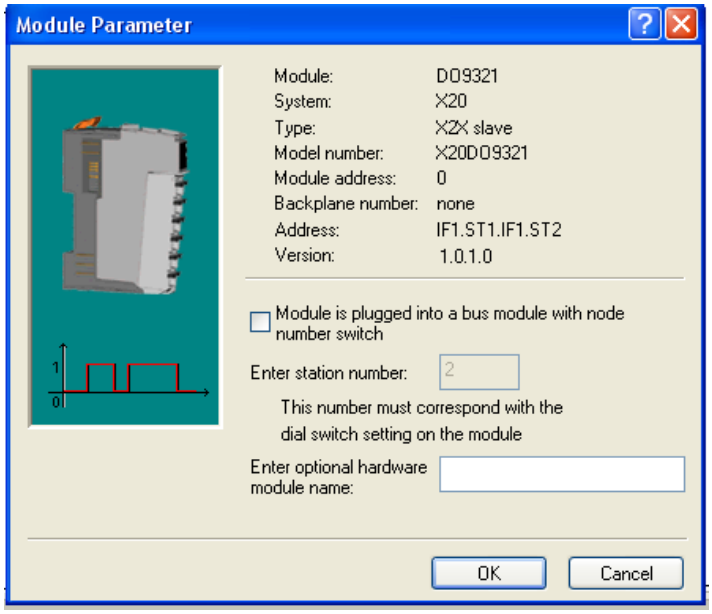


Figure 17: FieldBus DESIGNERNer DO9321 Module Configuration

The station number is set by default by the tool. It is very important to make exactly the same description as in the real hardware. Everything has to be in order, for instance, if the order in the physic hardware is the BusCONTROLER + Power Supply + Digital Input + Digital Output is essential to observe this order in the device description. It is compulsory to repeat this process to add all the cards we have in the project.

Once this work is done, and before building the project the last step is to set the X2X link properties (Figure 18). This window can be open right clicking on IF1 and after on **properties** but in the X2X thumb index..

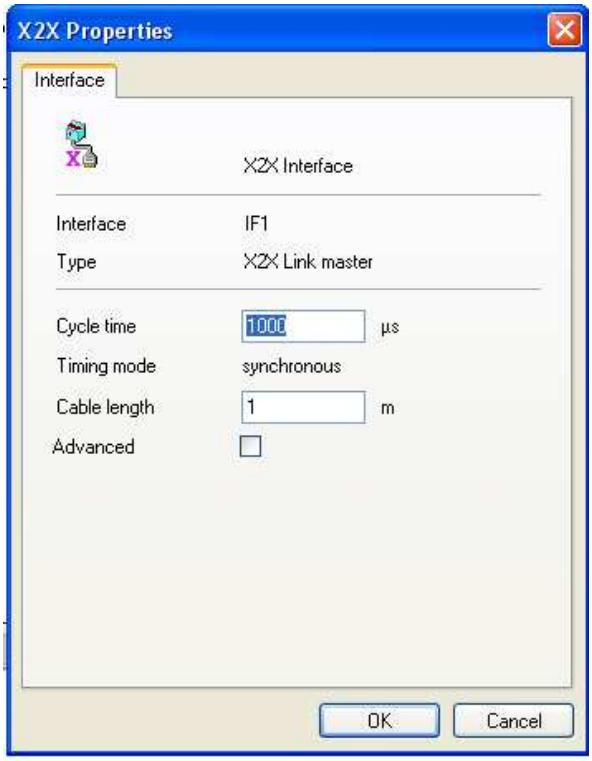


Figure 18: FieldBus DESIGNERNer X2X Properties

The first adjustable variable is cycle time of the X2X cycle. The X2X bus cycle time, is the cycle time that the bus controller has inside to refresh the variables of the cards, in order to send them after through the POWERLINK network. The cycle time determines the maximum number of modules permitted on the X2X Link. The shorter the cycle time the shorter the modules that can be added on the X2X link. It is supposed that the X2X cycle time has to be shorter than the Powerlink cycle time. The cable length is the distance between the modules. Is interesting to set it up when there is large distance between them. The last adjust possible it comes when the user click on advanced check box and in this way the user will be able to change the number of stations or I/O modules to be started by the bus controller. In this project this advanced option has not been touch so the configuration is the one that comes from default. After that click **Ok** to finish this process.

IMPORTANT: Do not mixed the two cycle times there are in the Network. The X2X cycle time is the one which the Bus controller has to refresh the values with the input and

output cards, and the POWERLINK cycle time is the one that the control node has with the computer in this case. So we can say that the POWERLINK cycle time is in a higher level.

2.4.1.5 Generating configuration files

The FieldbusDESIGNER generates the following description file in a folder named "Output" which is inside the Project folder. The user is asked at the time of creating the project where he wants to save the project.

XML Device Configuration (XDC): This file describes the devices and their configurations in standard XML format according to ISO 15745-4 as specified by the Ethernet POWERLINK Standardization Group in cooperation with the CAN in Automation (CiA).

To get this file is compulsory to build the program. This can be executed either using the menu option **Project / Build Configuration**, the F7 key, or the corresponding Build icon.

2.4.2 OpenCONFIGURATOR configuration

This section provides assistance for using openCONFIGURATOR, which will create the next necessary files to run the stack. The use of this tool is very easy because the programming interface of the tool is very visual. The two things that the user has to do are to attach the description files of the main node, computer, on one hand and in the other hand to attach the description files (XDC) of the controlled node. This description file of the controlled node is exactly the file that the user has recently created with the FiledbusDESINGNER. After building the project the tool will create you the necessary files to continue with the setting up of the network. The process of building a project will be divided in three steps.

2.4.2.1 Creating the project

Once the program has been installed and opened, the first step is to create a project. For this aim the user will select create a New Project and he will click OK. The next pop-up will appear (Figure 19) and this is the place where the user must set up the name of the project, the project directory and the save options. By default it is coming on Prompt which means that in this mode, the tool prompts the user asking, if the user wants to save the data or not before exiting from the screen. This is the configuration that comes by default.

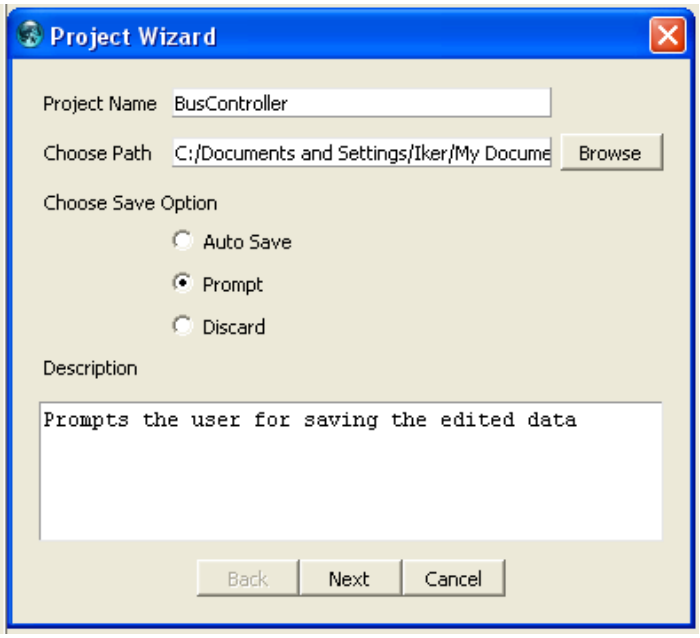


Figure 19: openCONFIGURATOR Project Wizard

Once the user has filled these gaps in the project wizard windows, the next pop up (Figure 20) appear and it is the one where the user has to set up the main features of the Main Node (MN) of the network. In our case the main node will be a Windows XP (32bits) computer so the user has to import the XDC/XDD file from this directory: *openPOWERLINKv_1.7/ObjDicts/CiA302-4_MN/00000000_openPOWERLINK_MN.xdd.*

The auto generate option it is really important as well, because in this way the PDO (Process Data Object) mapping of the MN and few objects of Controlled node (CN) will be generated automatically. This feature will be really useful and it makes the things easier because the tool creates the files automatically. Remember that, these files will be completely necessary for running the stack.

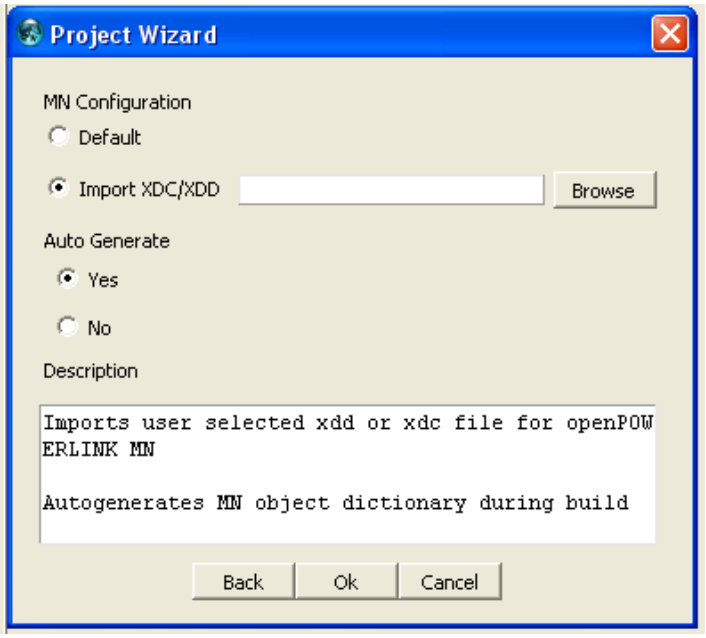


Figure 20: OpenCONFIGURATOR MN Configuration

2.4.2.2 Adding the controlled node and set the cycle time

Once the project has been created and the description (XDC) file of the MN has been added, the user will see the next window (Figure 21). The next step is to add the CN with is respective description file. For this aim the only thing the user has to do is right click on **openPOWERLINK_MN(240)** and then left click on **Add CN.** After a short while, the user will see (Figure 22) window and there he might change the next parameters. The name of the node, it is not really important because the only change is the visible name on the tree. The NodeID, here the user has to be really careful because has to be the same like is set it up on the real hardware. Apart from that the value must be between1-239. Finally the last thing that the user has to do is to import the XDC/XDD file for the application that can be founded on the CD attached to the report, following this path: *VUT\Project Folder\FieldBus Designer\A1\Output*. Otherwise the user can find the XDC file of the CN, exactly in <Project location> /<Project name>/cdp_xap folder. After that the user will press OK to finish with the adding process.

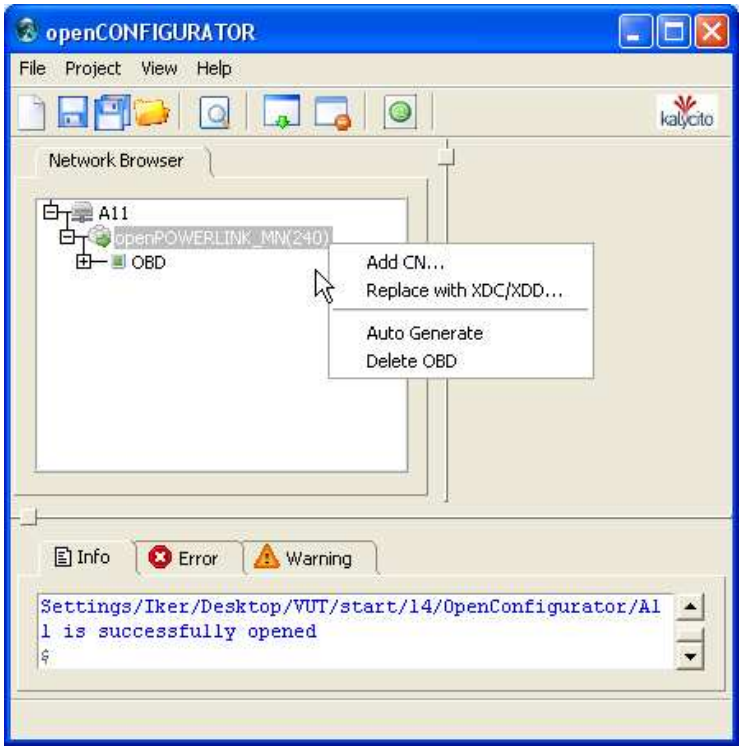


Figure 21: OpenCONFIGURATOR main window

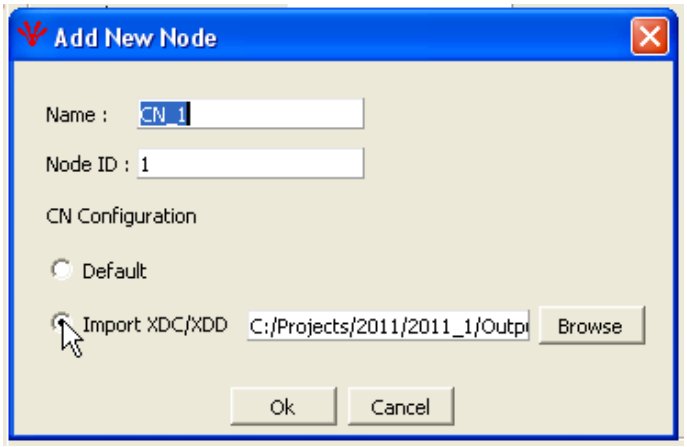


Figure 22: OpenCONFIGURATOR Add New Node

After that the user will have the project almost finished. The next and the last step will be to set up the cycle time. For this aim, it is interesting to change the view of the tree by clicking on **View** and then switch from normal view into advanced view. In this way

the user will be able to see all the objects of the MN and also all the objects of the CN. Here it is really important to set the cycle time of the POWERLINK network because this value will overwrite the one the user set up before, exactly on the FieldbusDESIGNER. There are two options to change the cycle time. The first one, if the user clicks just on openPOWERLINK_MN(240) a properties window will be shown on the right side of the screen where the third option is exactly the cycle time. The second one instead, is to modify the object 0x1006, which the user can find it clicking on the small '+' to see the long list of the MN OBD. Once the user has founded the object, the only thing has to do it, is to change the value, which can be found on the right side of the screen. In this project the cycle time is set to 10000µs, a suitable cycle time to control the Analog system. The openCONFIGURATOR also makes sure that the object 0x1006 is the same on all nodes in the network (otherwise the nodes would not work together)[12]. Is possible to check that if you check the value of the object 0x1006 of the CN you will se that is exactly the same as the one that has been set up on the other objects.

2.4.2.3 Generating configuration files

The openCONFIGURATOR tool generates five files. These files will be present in <Project location>/<Project name>/cdp_xap folder (Figure 23) or if you are trying to build the same application like this project, can be found on the CD following this path: *VUT\Project Folder\open CONFIGUARTOR\A11\cdc_xap*

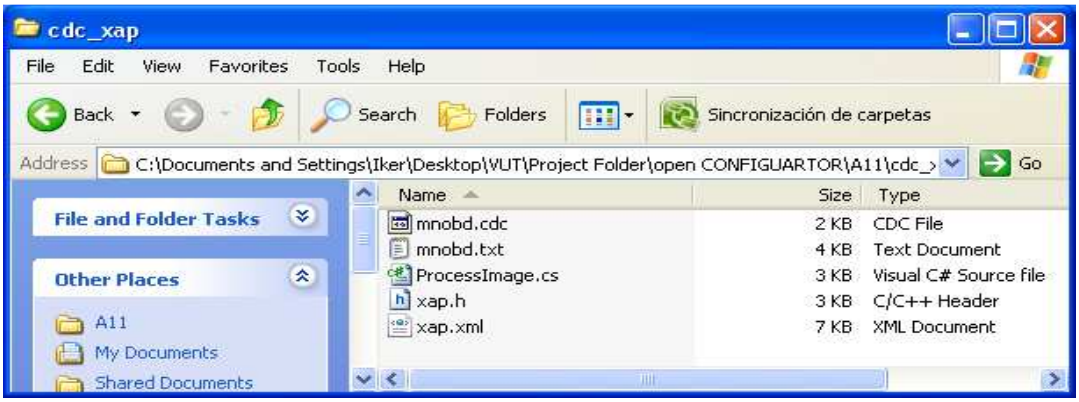


Figure 23: openCONFIGURATOR cdc_xap folder

Out of these five files, actually, four of them are really important files. The *mnobd.txt* file is a text version of the binary cdc. file. The *mnobd.cdc* file is the CDC binary file used with the stack. The *XAP.h* file is the header file for the application and finally the *XAP.xml* is the XML file with variables names, Datatype, datasize, ByteOffsets and BitOffsets. After the build, these files generated by the tool, shall be copied to the openPOWERLINK location. In this project exactly in the following path: VUT\Project Folder\openPOWERLINK_v1.7\Examples\X86\Windows\VC8\demo_cfm_pcap.

2.4.3 Selection of the system we want to control

The next step to continue with the setting up of the network, is to create a txt file which will contain information about which system we want to control and how we want to do it. In this project the user has the opportunity to choose between two different systems. Because of the high demand exist today on the market about controlling analog systems the project has been created to be able to control rather an analog system or a digital system. For this aim has been created a Visual Interface (Figure 24) to make things easier for the user.

The Visual Interface has been created using the Microsoft Visual Studio 2008, and it is designed only to make the setting up of the network easier, being much more useful and easy access for any kind of people even for the ones that they don't have to be necessarily experts on the programming field. The visual interface has been programmed in C++. As is said before this tool will create a file called 'SetupFile.txt' which will contain all the information about the system we want control. The use of this tool is actually really easy because it only has one window and all the user has to do after open it, is to choose which system they want to control. In the case of choosing the Analog system, the control of the wind tube in this project, the tool offers the user one position option which is related with the height of the cup into the model. If the user goes forward in this document will find a description of the model we control, as far as now, called analog system.

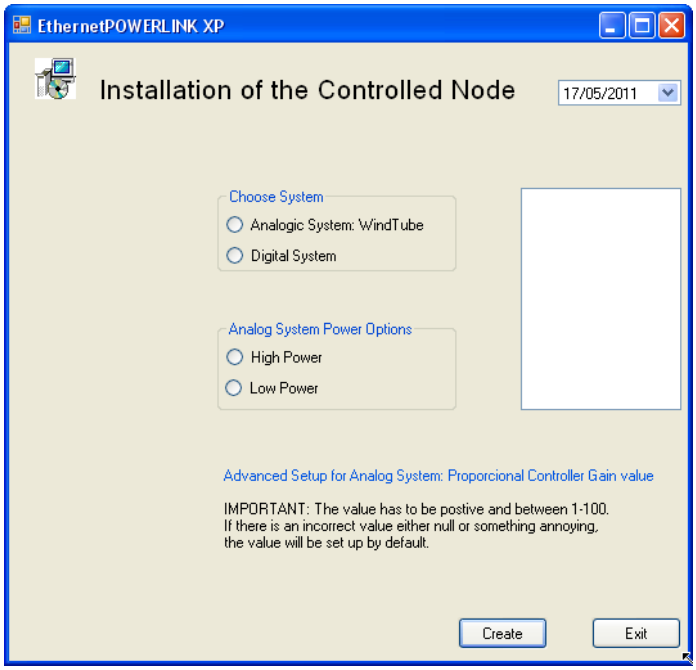


Figure 24: Visual Interface EthernetPOWERLINK XP

This visual interface generates one file. This file will be present in <Project location>/debug folder. In the CD:*VUT\Project Folder\WindowsFormApplication\Debug*

2.4.4 Adjust all the parameters to create a safety network

Before switch on our device it is neccesary to adjust all the parameters in our computer to create a safety network between the CN and the MN, which means, between the B&R developed Bus Controller and the XP computer. As we are going to connect both devices to each other we will create a net between them. Our project is controlled in real time so it is incredibly important to have a safe network to avoid all kind of problems related with networks [5]. To create a safety network it is compulsory to adjust some parameters on our computer network card and establish some changes on the internet protocol properties (Figure 25).

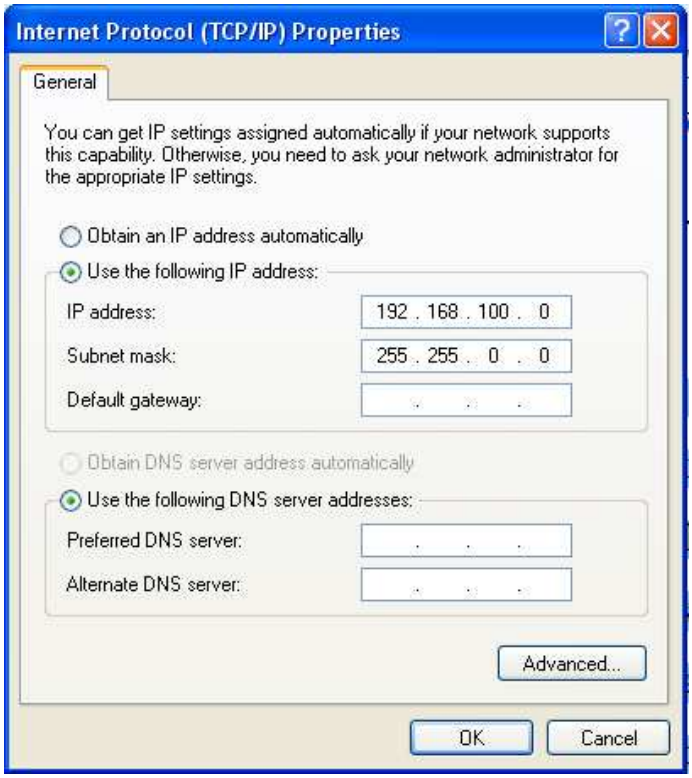


Figure 25: Internet Protocol (TCP/IP) Properties

The safest thing in this case is to create a private network. In the Internet addressing architecture, a private network is a network that uses private IP address space, following the standars set by RFC 1918 and RFC 4193 [13]. These addresses are commonly used for home, office, and enterprise local area networks (LANs). Usually the internet routers are configured in one way that they rull out any traffic that goes to the private IP address. This isolation makes the private networks more safeties, because it makes really difficult for someone from other network to be able to establish a direct connection to a machine or device through the network. This feature is due to it is no possible to establish a connection between different private networks through the internet. In the same time, more than one company may use the same addresses without being worried about the possible collisions with them.

These addresses are characterized as private because they are not globally delegated, meaning they are not allocated to any specific organization, and IP packets addressed by them cannot be transmitted into the public Internet. If such a private network needs

to connect to the Internet, it must use either a network address translator (NAT) gateway, or a proxy server.

They are different kinds of private IP addresses but the one used in this project it is the C Class network private address. The main difference between them is the range that has each of them, which has direct influence on the network properties. In this project as shown in the (Figure 25) we use the following IP address and Subnet mask:

IP address: 192.168.100.0

Subnet Mask: 255.255.0.0

NOTE: The network also works if we don't change any configuration but a warning message is shown on the computer screen. This warning message says that, the connections could have a limited connectivity or might be a wrong connection. As we need a reliable connection not to have problems with the real time communication this values are set it up to avoid any problem.

2.4.5 OpenPOWERLINK_v1.7

If the user is in this point of the report there are a few steps left and the powerlink network will be running. At this point the user is going to start making the last adjustments to set up the concrete network. This step is probably the last one but, it is the most important one because here is going to be where the user will program the code of the application he wants to control. Here there is all the Application Programming Interface (API) which has to be familiar with the user because here is going to be where he is going to set up our system. This project has been created from the folder is mentioned before which is openPOWERLINK_v1.7. This folder contains the Ethernet POWERLINK stack for the Managing Node (MN) and also the slaves Controlled Node (CN).

2.4.5.1 General description

In this folder there are more than one project examples, for windows and also for Linux but according to the devices used in this project, the most suitable project example is

the one is in this path: *openPOWERLINK_v1.7\Examples\X86\Windows\VC8\demo_cfm_pcap.*

This example project will be the start point of our application. Almost all the files the user has created in the previous steps have to be copied exactly in this folder. In fact this folder is going to be the project main folder. Here there is a small summary to know exactly which files have to be copied here. On one hand, there are the files created by openCONFIGURATOR, 5 files, and on the other hand we have the file that has been created with the visual interface. All the files have to be copied here.

There are more than one project examples but in this project the Main Node (MN) is a Windows XP (32bits) computer and the Controlled Node (CN) is B&R developed Bus Controller so this two features makes essential to build the project in VC8\demo_cfm_pcap.
Once all the files are copied in this folder the next step is to execute Microsoft Visual Studio 2008 and open the project following the steps as on the (Figure 26).

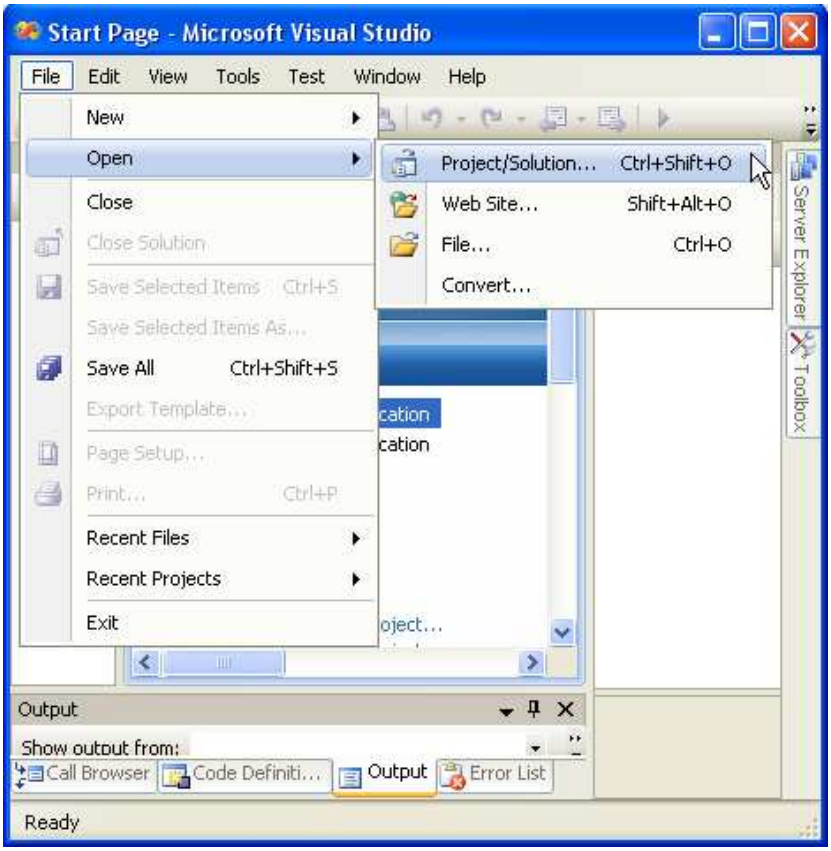


Figure 26: Microsoft Visual Studio Start Page

After opening the project the user has to make some light changes on the configuration. These changes will be applied on the 'demo.main.c' C file because the rest of the necessary files have been already manipulated by the openCONFIGURATOR. In this project there is all the necessary files and everything to run the stack. It is not an easy understanding project so this document only focuses on the essential points to adapt the program. As is said before in this report, is a project provided by SYSTEC electronics, which is a really reliable company in our industrial and technical environment. This report doesn't go further with the explanation of how the program works, if the user wants to go further with the project documentation [5] will be able to find further information.

The First essential change in our program is located into the global definitions. Here there is established the MN Node ID which is set up by default to 240 which is correct as you can checked on the openCONFIGURATOR (Figure 21). After this code line there

are two parameters that the user has to manipulate. These are the IP_ADDR and the SUBNET_MASK. In the point (2.4.4) of this report the user has already adjust some parameters with the aim of having a safety network, which are exactly these ones. So the only thing that the user has to do it, is to set up the same values as in the (Figure 25**¡Error! No se encuentra el origen de la referencia.**) making equal the IP_ADDR to 192.168.100.0 and the SUBNET_MASK to 255.255.0.0.

The Second essential change in the project has been the part that it is in charge of reading the information that contains the "SetupFile.txt". This file is created by the visual interface and it contains information about which system wants to control the user and how he wants to do it. In this report there is no further informaiton about how is programed this code. For further information here will find it, appendix [code part 1].

Finally, the third change, is probably the most difficult and the most inportant because it is exactly where the user program the control code of the system he wants to control. Almost at the botton of the 'demo_main.c' file the user will find a function called "tEplKernel PUBLIC AppCbSync(void)". This function is in charge of the control code. As every function in this example project, the first code line has to be "tEplKernel EplRet = kEplSuccessful;" to verify that the function has been properly opened. After this code line, the user can program whatever he wants but always using the concret functions either to tranfer the values from the computer to the Input / Output cards or get the value from them to the computer as well.

To know wich one is the sample time, it is essential to know that this function, "tEplKernel PUBLIC AppCbSync(void)" is called with each sent of SoC.Is it possible to check that further in the document (Figure 28). Everytime the SoC is sended through the POWERLINK network to sincronize all the devices this function is called. It makes that the frequency of the function is equal to the Powerlink cycle time.

When the user is programing the code has to take into acount that this function is executed cyclically. Apart form that it is essentyal to use this function" EplRet = EplApiProcessImageExchange(&AppProcessImageCopyJob_g);" to make the data tranfer possible. After these code lines, in this project there is all the code that controls both systems. On one hand there is the one which controls the Analog system with a proporcional controller and on the other hand there is the control code to control the

digital system. Actually, the digital system code is a really simple code, just to verify that there is a real time comunication between the Bus controller and the computer,, which would mean, that the network built by the user works properly. For further information of the control code the user can go appendix of the document [code part 2].

2.4.5.2 Program debbuging

The program is divided in three steps in this project and the third one depends on which system did the users choose before, exactly on the visual interface made exactly for this project, otherwise is divided in two. Actually, the project by itself it comes only divided in two steps but the third one has been added to offers to the users another option. As is said before in this document, after building the project the user have the chance to change between two different systems to control. If the Analog system is chosen, this process it is control thanks to a proportional controller which has some parameters which they are explained forward in this document. So this third step exists to give an opportunity to the user to manipulate one of these parameters. Now the report will go through the third steps. Then the user will be able to verify all this information in an example of a Windows command window (Figure 27).

2.4.5.2.1 First step of the program

The first step of the program is actually the optional one; it will appear only if the user has chosen to control the Analog system an inside it, if he has modified the power options (Figure 24). Only in this case the user will have the opportunity to change the proportional controller gain value. So this parameter is really interesting to check the system stability. So as soon as the windows command window is shown the first parameter to change will be the Proportional controller gain value.

2.4.5.2.2 Second step of the program

The second step of the project is exactly where the user has to select which network hardware is in charge of the communication with the Bus Controller. In this case the Bus Controller is connected to the computer through the Ethernet wire with the RJ45

connector so, it means that in this project the card that is in charge of the connection is the one called “Broadcom NetXtreme Gigabit Ethernet Driver” so all the user has to do it, is to choose the proper Ethernet card.

2.4.5.2.3 Third step of the program

The third part of the program is when the stack starts running which means that after recognizing the device the code that has been programmed will start running cyclically. It is possible to see all this in (Figure 27).

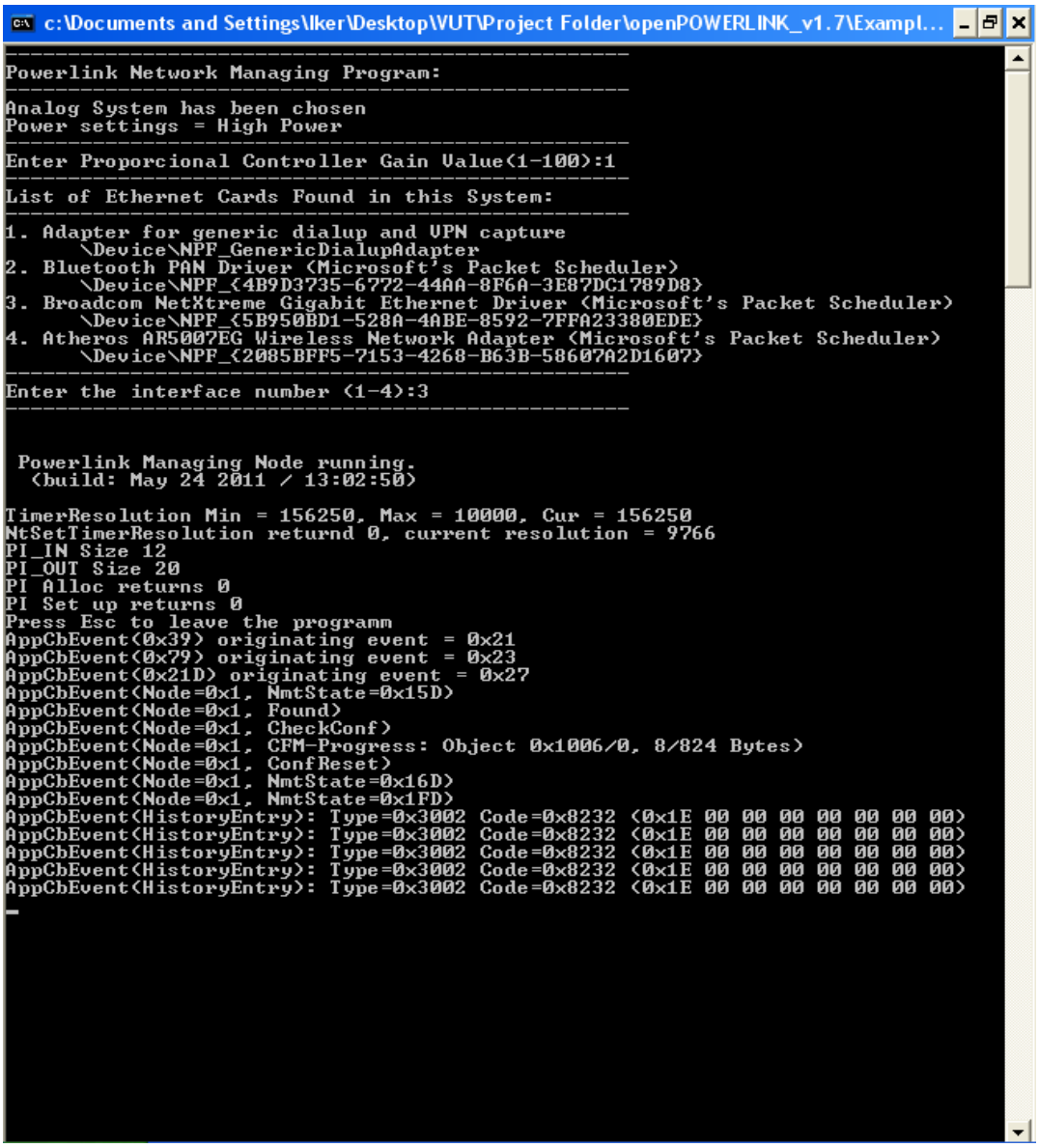


Figure 27: Cmd on the execution

Once the program is running it is really interesting to check how is it going the data exchange between the MN and the CN because it is a good way to learn and to see how often and how is done the communication. For this aim on the appendix of this document there is one picture that they verify that the cycle time set it before, it works properly. Here (Figure 36) the user will find this picture, they are like two one after the other, and it is essential to check what time were sent both. The date and time are underlined by a yellow square.

2.4.5.3 Sample-Time of the variables

In every controlled system there is one parameter that it is essential to know, to make sure that the communication is on real time. This parameter is the Sample time. If there is an application and it really needs to be in real time the user has to measure the sample time because it is closely related with the Sample time. The next (Figure 28) try to summarize which factors interferes in the sample time. The function where the control code stays is called with each sent of SoC. That features means that the sample time in this project is exacltly the same like the Powerlink cycle time.

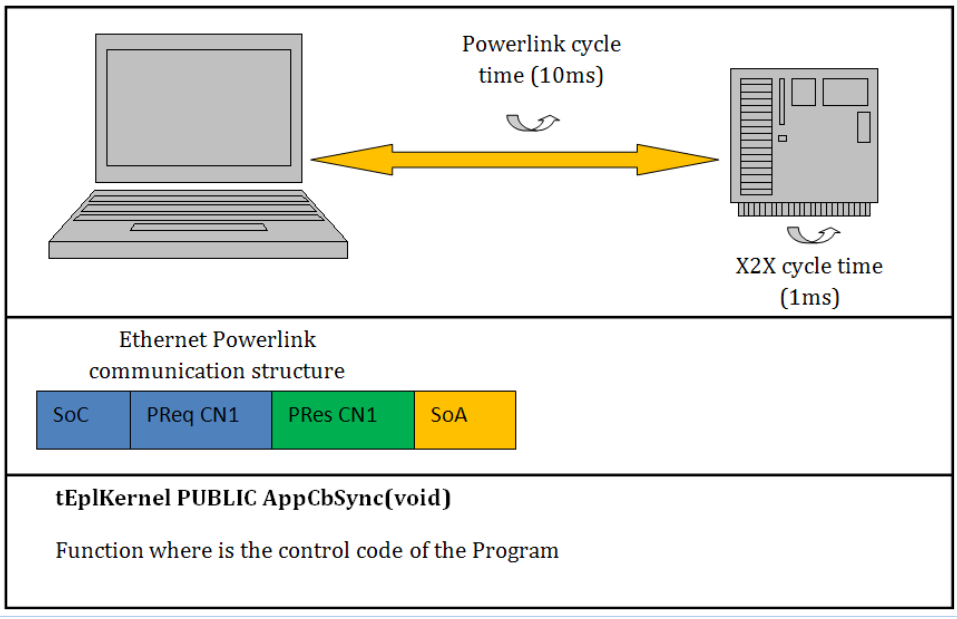


Figure 28: Sample time: Scheme of the project

2.5 Controlled systems

In this part of the document the explanations will focus on the systems used in this project, to verify that we achieve a real time communication. As it is said before in this report, the program is able to control two different systems.

2.5.1 Digital System

The digital system is really easy because it only switch on one output when the user pushes an input and it switch off the output when the user connects the other input. Here you can verify thanks to this diagram flux (Figure 29) what the program do. As you can say the programming code it is really simple, and it is due to, the aim of this project it wasn't to create some difficult programming code, it was to create some program to verify the real time communication. Apart from that it wasn't enough material to build some nice digital application so as you can see in the video that comes on the CD, in the last page of this report, the digital system is checked manually.

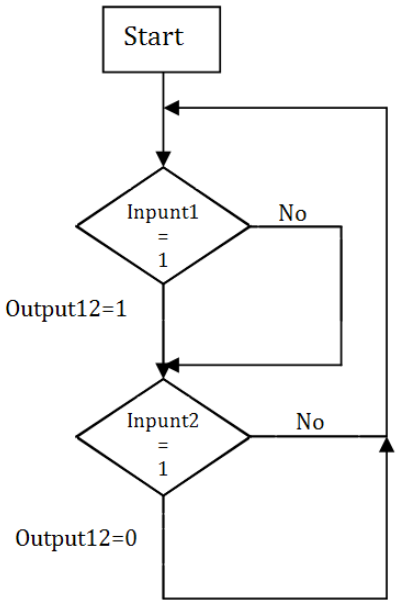


Figure 29: Diagram flux of the digital system

2.5.2 Analog System

The second control part of this project has been to control an Analog system which is basically the control of a Wind Tube. This system has one DC engine at the bottom of the tube and in the top of the tube has an ultrasonic sensor, which measures the distance to the bottom. Inside the tube there is a cup which will be moving inside the tube. The movement is thanks to the air flow created by the DC engine at the bottom of the model, so depending on the airflow the cup will go up, go down or will keep up the position in a concrete high. In the Real world, to keep the cup in one concrete position, is absolutely necessary to measure the position of the cup on the tube. In the (Figure 30) you can see how the model looks like.

Figure 30: Analog System model: Wind tube

2.5.2.1 Control of the model

For this aim nowadays there are a lot of sophisticate methods to control the system but in this project the simplest controller has been chosen (Figure 31) because as is said before the only aim of this, it is to check that there is a real time communication

between the main node and the controlled node. The features of this controller, as shown below, there are not really complicated but if the application is not really demanding it is absolutely enough. In this project the application, the model, might be really demanding but the aim of this project has not been to have a very precise control of the model, so that's why the decision to use a proportional controller was taken.

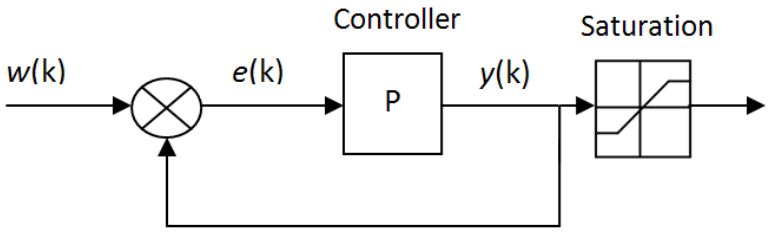


Figure 31: Proportional Controller block Diagram

In this project there has not had the necessity to multiply the result that comes from the controller with the plant, because it hasn't been necessary to control the analog system and check that the real time communications works properly. So the control code that has been implemented on the function previously said it is the next one.

1) $e(\mathrm{k}) = w(\mathrm{k}) - y(\mathrm{k})$
2) $y(\mathrm{k}) = \mathrm{K} * e(\mathrm{k})$
3) $y(\mathrm{k}) = \big(w(\mathrm{k}) - y(\mathrm{k})\big) * \mathrm{K} + \mathrm{x}$
4) $t = T_s * \mathrm{k}$

T_s = Sample time
t = time (s)
K = Gain
k = Discrette time

And then some limits are applied as well, to avoid that when saturation the levels goes higher than 10 V. As is said before the output level of the analog output is goes between -10 V and 10 V, but in this project the range used is between 0 – 10V. The analog input work exactly in the same way like the output and we also use only the positive range. This is due to the ultrasonic receiver gives us a value between 0 – 10 V.

IMPORTANT: As you will notice on the third equations there is an 'x' parameter. This parameter has been established by me to make the system more stable. It has been set up following the testing results to improve the behavior of the model.

2.5.2.2 Results

In this part of the document some results of several testing tries are shown thanks to some graphics. Actually there are two different tests. The first one is the most stable system has been achieved in this project (K=1) (Figure 32). Related with the position, the model has two stickers in different highs. The user also has the chance to manipulate them in this project. The second test has the particularity that the proportional controller gain has been establish to have an unstable system (K=5) (Figure 33).

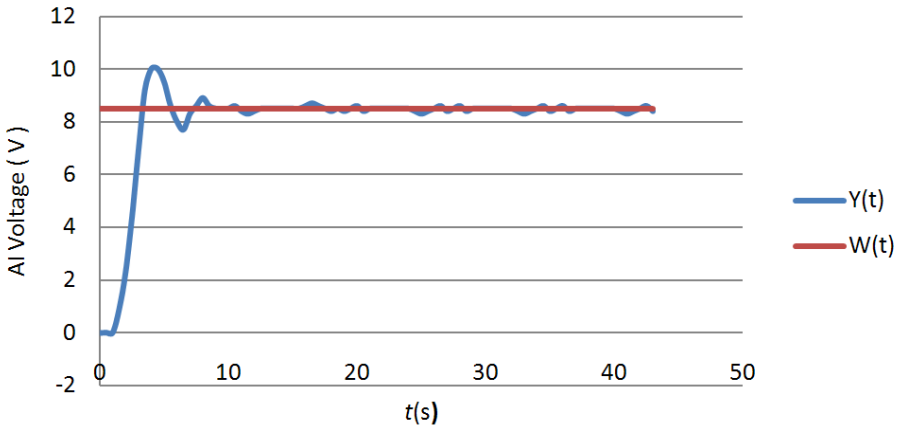


Figure 32: Wind Tube: Controller Gain (K=1)

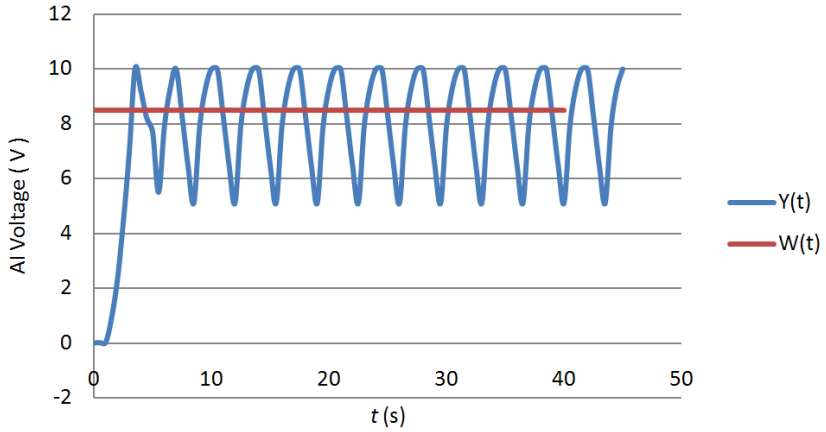


Figure 33: Wind Tube: Controller Gain (K=5)

3. Conclusion

This thesis gives an overview of the Ethernet Powerlink protocol and also shows the process of a complete installation of a Powerlink network. This network is created between a Bus Controller developed by B&R Automation and a Windows XP computer. At the beginning, an Introduction of the problems with the networks and his possible solutions are presented. The document also deals with the aims of this project and a exact schedule of how has been developed this project is shown.

In the second Chapter a summary of the development of the project is presented. This chapter of the document is the core of the thesis because here appears all the necessary knowledge to develop a system. This chapter is divided into 5 parts where each one explains essential knowledge of the project. On the first part the Etherenet Powerlink protocol is presented where all his main features are mentioned and also it is explained where this protocol comes from. On the second part there is a general description of the bus controller used in this project. The next part tries to summarize the main steps to get all necessary software to be able to put working the system. The fourth part goes program by program explaining all the steps like in a tutorial to set everything up. Finally the last part only summaries the basics of the scale model used in the project to check the real time communication.

Basically the goal of this thesis is to make as easy as possible for every single user the use of the X20BC0083 Bus Controller, and of course to give for professional programmers a light overview of what is a Powerlink Network and how can we build an application.

Appendix

1) First Semester Gant Diagram:

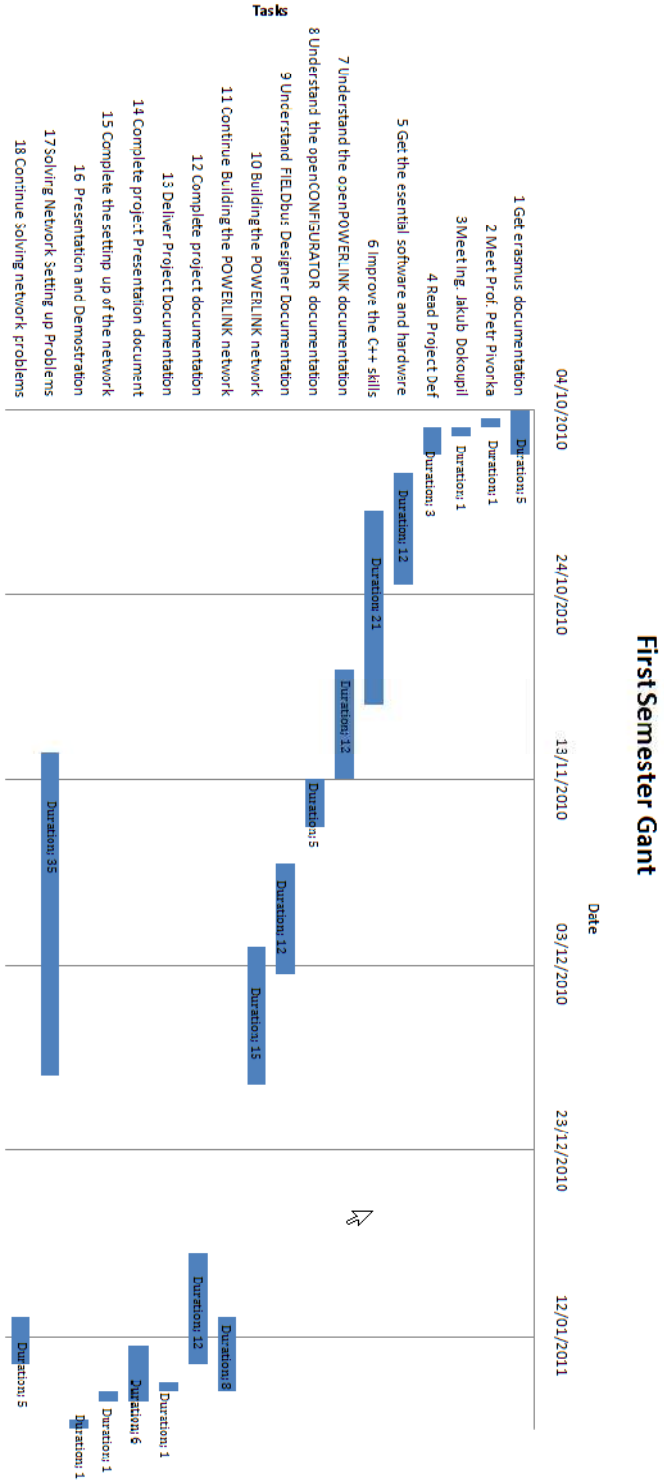


Figure 34: First Semester Gant Diagram

2) Second Semester Gant

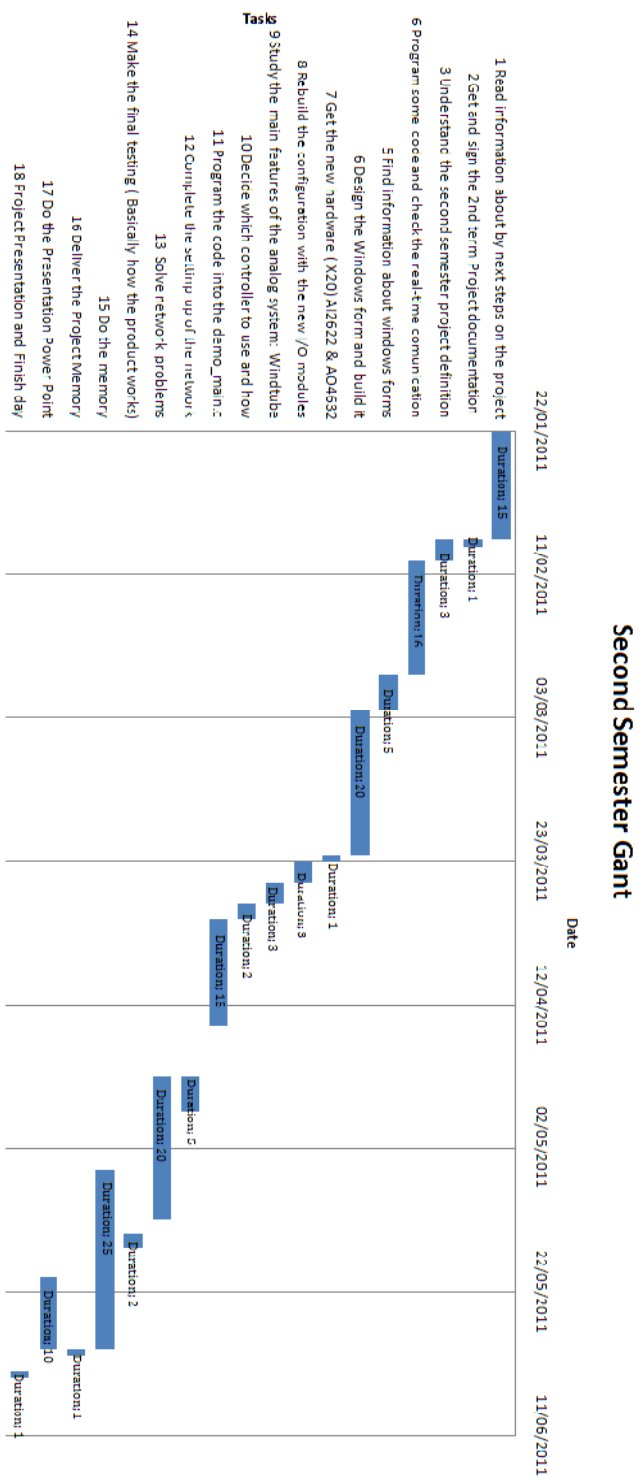


Figure 35: Second Semester Gant Diagram

3) Wireshark Checking Cycle-time:

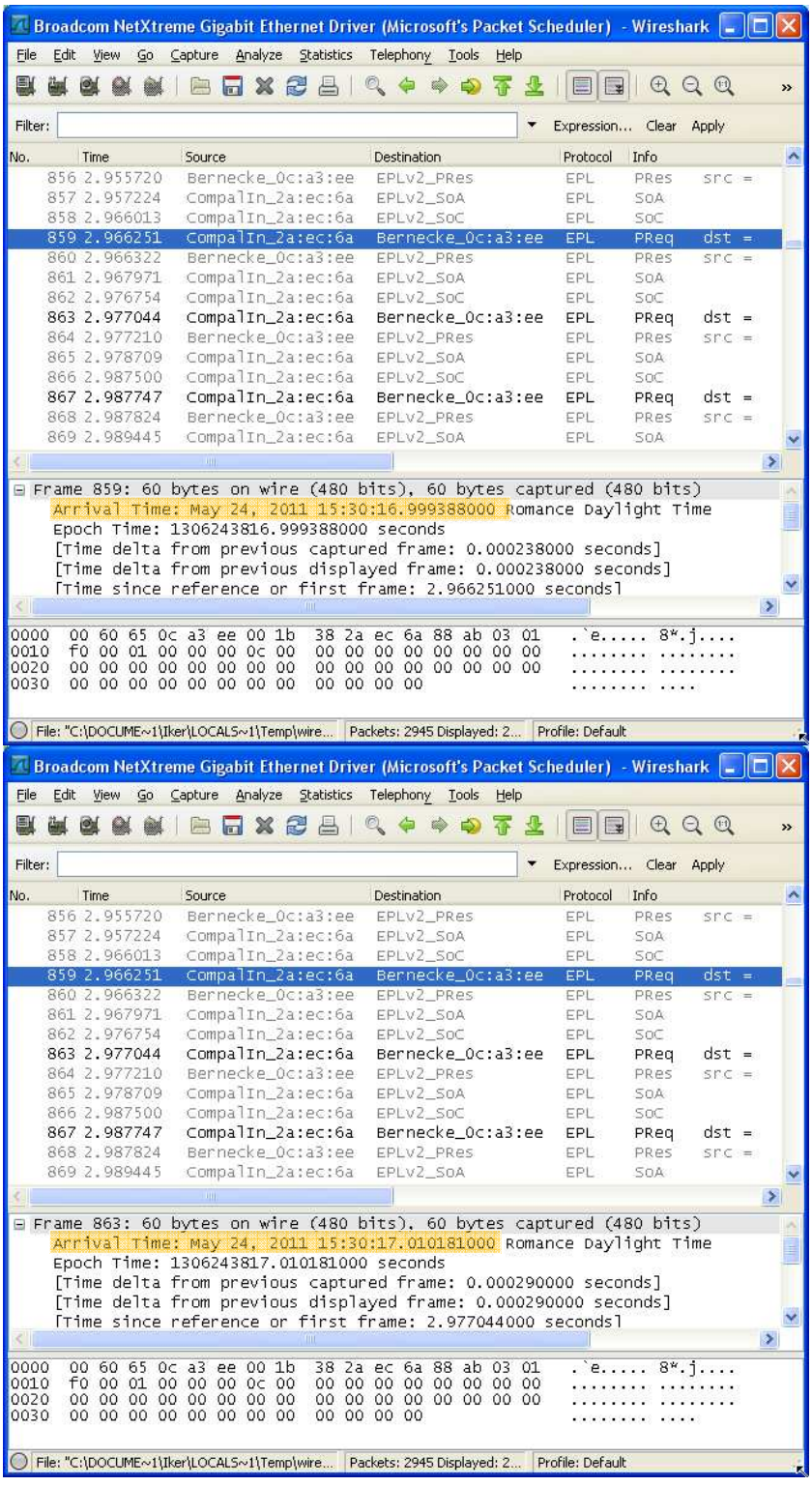


Figure 36: WireShark, Cycle time checking

4) demo_main.c program code part 1:

```
char c;
int num;
    FILE*fitxeroa;
    fopen ("SetupFile.txt" , "rt");
    if ( (fitxeroa=fopen("SetupFile.txt" ,"rt")) == NULL )
    {
            PRINTF0 ("Information");
    }
    else
    {
        c=fgetc(fitxeroa);

            if (c=='1'){
                  program=1;
                  PRINTF0 ("Information");
                  gain = 1;
                  x=7; }                    // High Power
            else if (c=='2'){
                  program=2;
                  PRINTF0 ("Information");}
            else if (c=='3')
            {
                  program=1;
                  PRINTF0 ("Information");
                  x=7;        // High Power parameter
                  PRINTF0("Gain Value(1-100):");
                  scanf_s("%f",&gain);
                  getchar();

                  if ((gain < 1) || (gain > 100))
                  {
                        PRINTF0("\nNumber out of range.\n");
                        goto Exit;
                  }

            }
            else if (c=='4')
            {
                  program=1;
                  PRINTF0 ("Information");
                  x=4;        // Low power parameter
                  PRINTF0("Controller Gain Value(1-100):");
                  scanf_s("%f",&gain);
                  getchar();

                  if ((gain < 1) || (gain > 100))
                  {
                        PRINTF0("\Number out of range.\n");
                        goto Exit;
                  }
            }
      }
```

5) demo_main.c program code part 2:

```
tEplKernel PUBLIC AppCbSync(void)
{
tEplKernel          EplRet = kEplSuccessful;
static unsigned int uiDigitalModData = 1;
float Y=0;        // Value applied to the Output
float r=0.0;      // Variable where stays the Readed Value in Volts
float W = 8.5;    // Desired position


if (program == 1)       // Analog System Running
{
      EplRet =
      EplApiProcessImageExchange(&AppProcessImageCopyJob_g);
      AppProcessImageIn_g.CN1_M01_X20DO9321_DigitalOutput01 = 1;
      r =
      (float)((AppProcessImageOut_g.CN1_M03_X20AI2622_AnalogInput01)
      *(10.0/0x7fff));
      Y = ((W-r)*gain)+x;
      if (Y > 10)
            Y =10;
      AppProcessImageIn_g.CN1_M04_X20AO4632_AnalogOutput01
      =(float)(Y*(0x7fff/10));
      }
else        // Digital System Running
{
      EplRet =
      EplApiProcessImageExchange(&AppProcessImageCopyJob_g);
if (AppProcessImageOut_g.CN1_M02_X20DI2377_DigitalInput01 == 1)
      AppProcessImageIn_g.CN1_M01_X20DO9321_DigitalOutput12 = 1;
if(AppProcessImageOut_g.CN1_M02_X20DI2377_DigitalInput02 == 1)
      AppProcessImageIn_g.CN1_M01_X20DO9321_DigitalOutput12 = 0;
}
    return EplRet;
}
```

*These pieces of code, has been lightly modified from the real ones, but the follow the same structure.

References

[1] Ethernet POWERLINK Standardization Group: Ethernet POWERLINK Basics,

http://www.ethernet-powerlink.org/

[2] SANCHEZ DIAZ, R. "CANopen" Departamento de Informática y automática, Facultad de Ciencias, Universidad de Salamanca,

http://gro.usal.es

[3] Datasheet " X20BC0083", accesible at

http://www.br-automation.com

[4] Datasheet "X20AI2622" accesisble at

http://www.br-automation.com

[5] openPOWERLINK: Ethernet POWERLINK Protocol Stack, Software Manual,

http://www.systec-electronic.com

[6] B&R Automation FieldbusDESIGNER program,

http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/cookies_allowed.htm?caller=services_168539_ENG_HTML.htm

[7] OpenCONFIGURATOR V1.2.0 program,

http://sourceforge.net/projects/openconf/

[8] Active TCL 8.5.1.9 driver,

http://www.activestate.com/activetcl/downloads.

[9] WinPcap Driver,

http://www.winpcap.org/install/default.htm.

[10] Dotnetfx35 driver,

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=d0e5dea7-ac26-4ad7-b68c-fe5076bba986.

[11] OpenPOWERLINK protocol stack,

http://sourceforge.net/projects/openpowerlink/.

[12] openPOWERLINK high level manual

http://www.kalycito.com/manuals_kalycito.php

[13] Ip Address Wikipedia,

http://en.wikipedia.org/wiki/Private_network

*All the documentation files that has been neccesary to build this project will be available on the CD.